INFORMATION AND CONTROL 45, 68–89 (1980)

Derivation Structures for Strictly Context-Sensitive Grammars

JOHNSON M. HART

Department of Computer Science, 915 Patterson Office Tower, University of Kentucky, Lexington, Kentucky 40506

A graphical technique is developed to represent derivations in strictly context-sensitive grammars, where the context symbols are not rewritten. The resulting context-sensitive syntactical graphs represent an equivalence class of derivation sequences, and the canonical derivation can be obtained from a well-defined pre-order traversal of the context sensitive syntactical graph. The advantage of context-sensitive rules over monotonic rules in certain cases is shown, and the desirability of mixed grammars is also mentioned. The context-sensitive syntactical graphs and their corresponding canonical derivation sequences extend the techniques used for type 0 derivations.

INTRODUCTION

The techniques for specifying the structure of derivations in phrase structure (Chomsky) grammars are well understood, at least when regular (type 3), context-free (type 2), or unrestricted (type 0) grammars are considered. For instance, derivation trees have long been used to describe derivations in context-free grammars. In addition, a variety of equivalent more general techniques exist to describe derivations in type 0 grammars, including the use of categories (Hotz, 1966), syntactical graphs (Eickel and Loeckx, 1972, and Buttelman, 1975), derivation words (Hart, 1976), and combinations of all of these (Hart, 1975; Révész, 1977). There is a problem, however, with describing derivations in type 1 (context-sensitive) grammars, especially when the "strictly" context-sensitive grammars are considered. The strictly context-sensitive productions used here are those of the form $\alpha A\beta \rightarrow \alpha X\beta$, instead of merely monotonic productions.

Initially, the existing techniques for describing context-sensitive derivations will be discussed, along with the difficulties associated with each. These difficulties are resolved in Section 2, where the concept of a syntactical graph is extended to show the context-dependencies in the sequence of production rule applications. The resulting context-sensitive syntactical graphs are developed more formally in Section 3. A unique pre-order traversal of these graphs is also shown

0019-9958/80/040068-22$02.00/0

to exist, and this traversal enables one to compute the canonical derivation of a class of equivalent derivations in the strictly context-sensitive grammar.

The canonical derivations are defined in Section 4 and are shown to be similar to those in type 0 grammars, but with some essential distinctions. Section 5 examines the relation between monotonic and strict CS derivations and proposes a form of mixed CS grammar.

Chapter 6 indicates some other aspects of the context-sensitive derivation structures, including a correspondence with permutations.

1. Strictly Context-Sensitive Grammars and Their Derivations

A context-sensitive grammar is generally written as

$$G = (V, \Sigma, P, S),$$

where V is a finite alphabet, $\Sigma \subset V$ is the *terminal alphabet*, $S \in V - \Sigma$ is the designated *start symbol*, and P is a finite set of production rules. By convention, we call $N = V - \Sigma$ the set of *nonterminal symbols*. Each production rule is of the form

$$\alpha \rightarrow \beta \qquad (\alpha, \beta \in V^+),$$

where, for CSG's, one of two alternative restrictions is applied; either:

(a) For all $\alpha \rightarrow \beta \in P$, $|\alpha| \leqslant |\beta|$,

where $|\alpha|$ refers to the length of string α, or

(b) Each member of P is of the form

$$\alpha A \beta \rightarrow \alpha X \beta,$$

where $\alpha, \beta \in V^*$, $A \in N$, and $X \in V^+$.

The first form is often referred to as being "monotonic" ("type 1," "length increasing," etc.), and CSG's in this form will be of no further interest since their derivations can be described as for type 0 grammars. That the two forms are equivalent is well known (Salomaa, 1973 and Harrison, 1978), at least, for any CSG in one form, there is a CSG in the other form generating the same language. Thus, the forms are *weakly equivalent*. This point will be discussed in Section 5.

A grammar in the second form is said to be a *strictly context-sensitive grammar* (SCSG). Thus, A can be rewritten as X only when it appears in the left and right context of α and β, respectively. A CS production rule of the form

$$\alpha A \beta \rightarrow \alpha X \beta$$

is often written as

$$A \rightarrow X \mid \alpha\_\beta.$$

We will generally use this second form in what follows to emphasize the use of context in the application of a context-free production. In addition, each production will be given a name (as in Hart, 1975 and elsewhere), so the form of a production will be:

$$\pi\colon A \rightarrow X \mid \alpha\_\beta$$

(*Note*: There is no uniform vocabulary in the literature to distinguish monotonic and strictly context-sensitive grammars.)

The most direct method of representing a derivation in a CSG is *generatively* as a sequence of derived sentential forms, that is, by:

$$\begin{aligned} S &\Rightarrow u_1\alpha_1 A_1\beta_1 v_1 \Rightarrow \cdots \Rightarrow u_i\alpha_i A_i\beta_i v_i \\ &\Rightarrow \cdots u_n\alpha_n A_n\beta_n v_n \Rightarrow w = u_n\alpha_n x_n\beta_n v_n\,. \end{aligned}$$

Here, $w \in V^*$ is the derived word (we are interested in derivations in general and do not insist that $w \in \Sigma^*$). For each i, there is a production $\pi_i : A_i \rightarrow x_i \mid \alpha_i\_\beta_i$ such that $u_i\alpha_i x_i\beta_i v_i = u_{i+1}\alpha_{i+1}A_{i+1}\beta_{i+1}v_{i+1}$. A more compact method of representing a derivation generatively is by a sequence of pairs

$$(\pi_1, l_1), (\pi_2, l_2), \ldots, (\pi_n, l_n),$$

where π_i is the name of the production applied at the ith step and $l_i = \mid u_{i-1}\alpha_{i-1} \mid$ gives the location of the rewritten symbol, with $l_1 = 0$.

The generative form of a derivation has several drawbacks, including lack of compactness and a failure to show the actual structure of the derivation in the way that a derivation tree or syntactical graph does. Furthermore, inessential changes in the order of rule application lead to equivalent derivation sequences.

Syntactical graphs and derivation words compactly represent a class of structurally equivalent derivation sequences, and hence eliminate the problems of the generative form. The use of syntactical graphs leads to the concept of a *canonical derivation sequence*, and the resulting structures lead to straightforward concepts of ambiguity, parses, and semantics defined on a derivation. When a derivation graph is drawn for a derivation sequence in a CSG, however, problems result immediately from the fact that spurious structural ambiguities are introduced since the idea of a syntactical graph requires that the context symbols be rewritten. For instance, consider a grammar with the productions:

$$\begin{aligned} \pi_1 &: S \rightarrow aAbBc \\ \pi_2 &: A \rightarrow x \mid a\_b \\ \pi_3 &: B \rightarrow y \mid b\_c. \end{aligned}$$

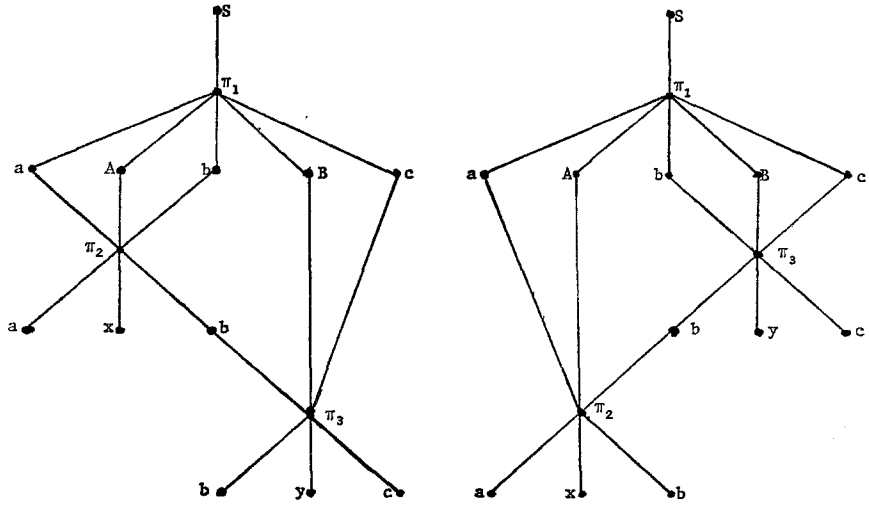


FIG. 1. Two distinct syntactical graphs of the word *axbyc* resulting from the rewriting of the context symbols.

There are then two distinct syntactical graphs deriving word *axbyc*, as shown in Fig. 1. These two structurally distinct derivations surely are not really distinct from the point of view of the strictly CSG, however, so some other concepts will be needed to describe CS derivations.

The example above can be extended to show that for any $n \geqslant 2$, there is some CSG with $n!$ distinct syntactical graphs for a single word. Buttlemann (1975) has also pointed out the existence of multiple "phrase structures" for a single derivation in a CSG.

The difficulty with syntactical graphs is due to the fact that the graph shows the context symbols as being rewritten, whereas they are actually used merely to permit the application of a context-free production. Peters and Ritchie (1969) used this concept to define an analysis on context-free derivation trees based upon the underlying CFG. The proper analysis is, however, not powerful enough, and a context-free string language still results. Joshi and Levy (1977) extended this result by defining "local transformations" on trees, but CFL's still resulted. The reader may find it interesting to use one of the popular grammars for $\{a^n b^n c^n \mid n \geqslant 1\}$ (in strictly CS form) and note how the proper analysis of Joshi and Levy permits spurious derivations in which symbols are used for context after they have been rewritten. It should be noted that the intent of Joshi and Levy was to analyze the linguistic use of context-sensitivity under certain circumstances, and this use does not increase generative power. Nonetheless, such use of context is useful from a linguistic point of view.

The next section will show how the concept of a proper analysis can be strengthened to give a precise description of CS derivations.

2. The Context-Sensitive Dependencies and Context-Sensitive Syntactical Graphs

Consider the CSG below in which S is the start symbol and we are not concerned with terminal and nonterminal symbols.

$$\pi_1 : S \rightarrow ABC$$
$$\pi_2 : A \rightarrow XY$$
$$\pi_3 : B \rightarrow DE \mid A\_$$
$$\pi_4 : E \rightarrow AA \mid \_\, C$$
$$\pi_5 : C \rightarrow FG$$

One derivation sequence in this grammar is (where the rewritten symbol is underlined):

$$Z \overset{\pi_1}{\Rightarrow} A\underline{B}C \overset{\pi_3}{\Rightarrow} AD\underline{E}C \overset{\pi_4}{\Rightarrow} ADAA\underline{C}$$
$$\overset{\pi_5}{\Rightarrow} \underline{A}DAAFG \overset{\pi_2}{\Rightarrow} XYDAAFG.$$

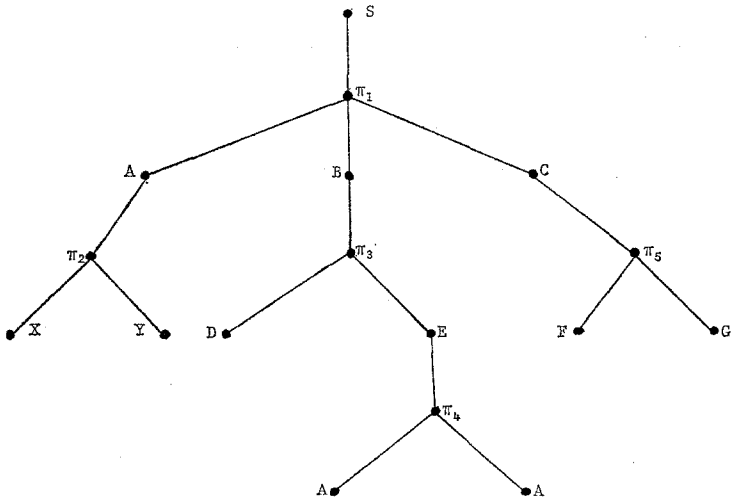


FIG. 2. A context-sensitive derivation shown as a derivation tree.

Figure 2 shows this derivation as a tree in the underlying CFG, and, of course, further analysis of this tree is required to confirm that it corresponds to a valid derivation sequence in the CSG. Note that in the derivation sequence, the application of π_2 can occur before that of π_5 or even π_4, but it cannot preceed the application of π_3.

Confirming that a tree corresponds to a valid derivation sequence in the CSG is not as easy it might seem, for Stradel (1978) has shown that this problem

(called "renormalization") is *NP*-complete, and membership in a CSL is *P*-space complete (Karp, 1972).

Figure 2 shows a certain relationship between the nodes of a labelled tree. We will denote by $<_r$ the relation between a node and its immediate descendant in the tree. We use the subscript r to indicate the relation is induced by a "rewrite" rule. Thus, if the tree is constructed from a set of labelled nodes where l: $N \to V \cup P$ is the node labelling function, then, the relation $<_r$ is a subset of $(l^{-1}(V) \times l^{-1}(P)) \cup (l^{-1}(P) \times l^{-1}(V))$. Taking the transitive closure is reserved as a later step.

In keeping with the idea of a proper analysis (Peters and Ritchie, 1969; Joshi and Levy, 1977), the next step is to show the "context dependencies" in the tree. That is, draw an edge from the symbols used as context to the production which uses that context. These edges are shown in Fig. 3 as dashed lines, and the relation will be denoted by $<_c$ for context dependency.

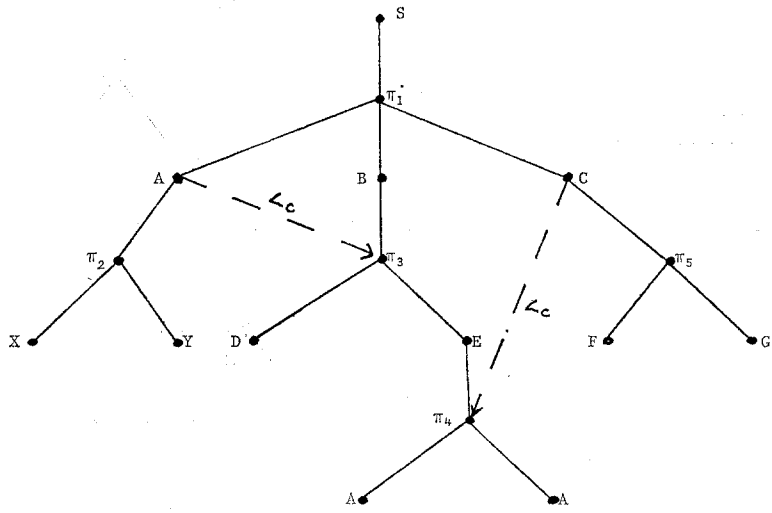


FIG. 3. The derivation of Fig. 2 shown with context dependencies.

Knowledge of proper analysis indicates that something is still missing. In fact, if the graph of Fig. 3 is sorted topologically by the methods indicated in Hart (1976) (i.e., visit the nodes as in a pre-order tree traversal but do not visit a node before visiting all of its ancestors with respect to $<_r \cup <_c$), Fig. 3 would indicate that the π_2 could be applied in the derivation sequence before π_3. This is not correct, however, because Fig. 3 fails to consider the fact that the node labelled A is not "free" to be rewritten until its last use as a context symbol. We now create the relation $<_f \subseteq l^{-1}(P) \times l^{-1}(P)$ to indicate that one production can be used only after all uses of that symbol for context by other productions. Figure 4

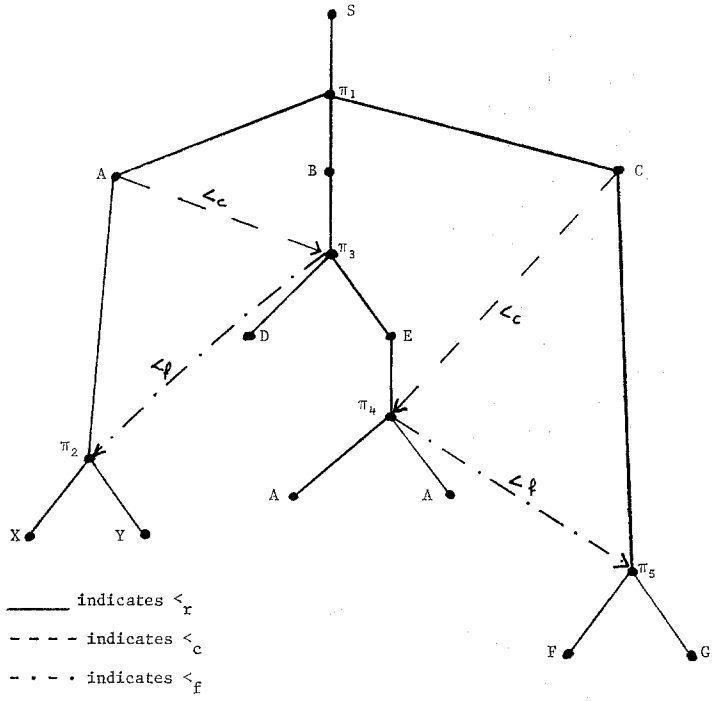


FIG. 4. A complete context-sensitive syntactical graph for the derivation in Fig. 2.

shows the resulting complete "Context-Sensitive Syntactical Graph" (CSSG) with $<_f$ indicated by dotted–dashed lines. The addition of $<_f$ now enables a traversal of the graph in such a way that the production nodes are visited in an order which corresponds to one of the equivalent derivation sequences, namely (in this example) $\pi_1\pi_3\pi_2\pi_4\pi_5$. The resulting derivation sequence will be defined as the *canonical derivation sequence* in keeping with the canonical sequences of type 0 derivations (treated from various points of view in Hotz, 1966, Griffiths, 1968, Hart, 1975, and elsewhere).

In a CSSG, we define $<$ as the transitive (but not reflexive) closure of $<_r \cup <_c \cup <_f$; i.e.,

$$< = (<_r \cup <_c \cup <_f)^+.$$

Harrison (1978, pp. 23–24) and others have noted that the left-to-right order of nodes in a derivation tree is as important as the rewrite dependency relation (called B for "below" in the discussion that follows). We call this relation L (left-of), and L is also defined to be transitive but not reflexive. Harrison also notes that for any pair of distinct nodes $x, y \in N$ (nodes of a derivation tree),

exactly one of the relations B, B^r, L, or L^r holds ($B^r = \{(x, y) \mid (y, x) \in B\}$ is the reversal of the relation). In fact, for any derivation tree with node set N,

$$\{B, B^r, L, L^r, =\}$$

forms a partition of $N \times N$. Alter and Hart (1979) have shown that this property extends to syntactical graphs as well, and we wish to define CSSG's in a formal manner so as to have this property.

Before defining CSSG's in terms of derivation sequences, we examine several of the consequences and advantages of this partition of $N \times N$ in the next section.

3. Doubly Ordered Graphs and Their Application to Context-Sensitive Syntactical Graphs

In keeping with the notation of Alter and Hart (1979), we have

Definition 3.1. Let N be a finite set and B and L be transitive relations on N such that

$$\{B, B^r, L, L^r, =\}$$

is a partition of $N \times N$. Then

$$\Gamma = (N, B, L)$$

is a *Doubly Ordered Graph* (DOG). A *labelled DOG over alphabet* V is a DOG together with a function l: $N \to V$ and is denoted

$$\Gamma_l = (N, B, L, l).$$

A DOG is a *DOG-tree* if for all $(x, y) \in L$ there is no z such that $(z, x) \in B$ and $(z, y) \in B$.

Note. A DOG-tree can have several roots (and is thus really a "forest"). B can be thought of as "below" and L as "left-of."

It is shown in the above reference that every syntactical graph is a labelled DOG, and, of course, every CF derivation tree is a labelled DOG tree.

A node $x \in N$ is said to be *B-most* if there is no $y \in N$ such that $(y, x) \in B$. L-most, B^r-most, and L^r-most nodes are also referred to. Likewise, we define the *immediate* relation. If A is a relation, the *immediate A relation* is:

$$A^0 = \{(x, y) \in A \mid z \neq x \,\&\, z \neq y \Rightarrow (x, z) \notin A \quad \text{or} \quad (z, y) \notin A\}$$

This is often referred to as the *transitive reduction* of A. The notation xAy is occasionally used if $(x, y) \in A$, as is xA^0y.

Since syntactical graphs represent phrase structure derivations, it is convenient to define the domain (start word) and codomain (derived word) of a labelled DOG, in keeping with the categorical approach. The following definitions are shown to be precise in Alter and Hart (1979).

DEFINITION 3.2. Let $\alpha = (N, B, L, l)$ be a labelled DOG. If $\{y_1, y_2, ..., y_k\}$ is the set of all B^r-most nodes in N such that

$$y_1 L y_2 L y_3 L \cdots L y_k$$

then the domain of α, denoted $\mathrm{dom}(\alpha)$, is

$$\mathrm{dom}(\alpha) = l(y_1 y_2 \cdots y_k).$$

If $\{z_1, ..., z_m\}$ is the set of B-most nodes in N such that

$$z_1 L z_2 L z_3 L \cdots L z_m,$$

then the *codomain of* α, denoted $\mathrm{cod}(\alpha)$, is

$$\mathrm{cod}(\alpha) = l(z_1 z_2 \cdots z_m).$$

The definition above is motivated by the fact that if α represents a syntactical graph, the derivation is from word $\mathrm{dom}(\alpha)$ to $\mathrm{cod}(\alpha)$.

The concept of a DOG is important because it allows us to define a generalization of the pre-order traversal of trees. It is this traversal that leads to the derivation words and canonical derivations of Hart (1975).

DEFINITION 3.3. Let N be a set with B and R transitive relations on N. A *pre-order traversal* (POT) of $\Gamma = (N, B, L)$ is a one-to-one onto function:

$$\rho: N \to \{1, ..., \#N\}$$

such that

(1) $(a, b) \in L \Rightarrow \rho(a) < \rho(b)$,

and

(2) $(a, b) \in B^r \Rightarrow \rho(a) < \rho(b)$.

Note. This definition requires B and L to be irreflexive. Otherwise put, ρ is a topological sort which satisfies both L and B^r.

LEMMA 3.1. *Let $\Gamma = (N, B, L)$ be a system as in Definition 3.3 such that $N \times N$ is equal to $B \cup B^r \cup L \cup L^r \cup =$. Then there is at most one pre-order traversal of Γ.*

Proof. Let ρ and τ be two distinct POT's of Γ. Let j be the smallest integer such that $\rho^{-1}(j) \neq \tau^{-1}(j)$, where $j = \rho(a) = \tau(b)$ for some $a, b \in N$. Now, $\tau(a), \rho(b) > j$. Therefore $(b, a) \notin L \cup B^r$ and $(a, b) \notin L \cup B^r$, which contradicts the assumptions of the lemma.

LEMMA 3.2. *Let $\Gamma = (N, B, L)$ be a system as in Definition* 3.3. *If Γ has a POT, then $B \cap B^r = L \cap L^r = L \cap B = L^r \cap B^r = \varnothing$.*

Proof. The existence of an element in any of the four intersections above would contradict the definition of a POT.

Now, the POT of any DOG can be easily computed by the following algorithm, and Lemma 3.1 assures us that this POT will be unique.

ALGORITHM 3.1. Construction of the POT of a DOG.

Input: $\Gamma = (N, B, L)$, a DOG.

Output: ρ: $N \to \{1,..., \#N\}$, a POT of Γ.

Algorithm:

(1) *Set* $N_1 = N$;

(2) LOOP: *Do* $i = 1$ *to* $\#N$;

(3) *Set* $a =$ the L-most, B^r-most member of N_i;

(4) *Set* $\rho(a) = i$; *Set* $N_{i+1} = N_i - \{a\}$; *end* LOOP;

(5) *Halt*;

The node $a \in N$ in Step 3 always exists uniquely. If $n = \#N$, the obvious brute force approach to implementing Step 3 has time complexity $0(n^2)$ giving Algorithm 3.1 a time complexity of $0(n^3)$.

The use of doubly ordered graphs as an abstract data structure for representing records in a data base has been developed in Hart (1979). The DOG is represented by a structure known as a "doubly keyed list" which can be traversed in linear time.

Algorithm 3.1 would also work even if $L \cap B^r$ were not null (hence $L^r \cap B \neq \varnothing$). Therefore, a sufficient condition for a unique POT (assuming transitive relations) is that

(a) $B \cup B^r \cup L \cup L^r \cup =$ is equal to $N \times N$,

(b) $B \cap B^r = L \cap L^r = L \cap B = \varnothing$ (hence B and L are irreflexive).

Nonetheless, in examining CSSG's, we will work with DOG's in order to establish a correspondence with permutations in Section 5.

Finally, the definition of POT in Definition 3.3 actually has three other variations (of the relations between B, L, and ρ) which would correspond to other types of tree traversal, such as post-order traversal.

Note that Algorithm 3.1 implies that, for any DOG, $B \cup L \cup =$, $B^r \cup L \cup =$, etc. are total orders on N.

We wish to define CSSG's in such a way that they have an algorithmically computable pre-order traversal with respect to the total dependency relation $<$ (described informally at the end of Section 2) and an appropriately defined "left-of" relation (to be called $\vdash$). To do this, CSSG's are defined inductively and are shown in this way to be DOG's.

In the process of giving a formal definition of CSSG's, the domain and codomain of the underlying tree of the CFG is used (B and L are the tree relations). $<$ and $\vdash$ are not included in the definition, as $<_c$ will be seen to be sufficient information. The definition constructs CSSG's inductively from derivation sequences. The derivation can be from any word, not just from the single start symbol.

DEFINITION 3.4. Let $G = (V, \Sigma, P, S)$ be a SCSG. The set of *context-sensitive syntactical graphs of* G is denoted by CSSG(G) and is defined to be a set of systems

$$\text{CSSG}(G) = (N, B, L, <_c, l),$$

where (N, B, L, l) is a labelled DOG tree over V and $<_c \subset N \times N$ such that CSSG(G) is the smallest set for which:

(1) $(\{n_1, \ldots, n_k\}, \phi, L, \phi, l) \in \text{CSSG}(G)$ where

$$L = \{(n_i, n_j) \mid 1 \leqslant i < j \leqslant k\} \quad \text{and} \quad l: \{n_1, \ldots, n_k\} \to V.$$

(2) If $\Gamma = (N, B, L, <_c, l) \in \text{CSSG}(G)$ with

$$\text{cod}(N, B, L, l) = u\alpha A\beta v (u, \alpha, \beta, v \in V^* \quad \text{and} \quad A \in V - \Sigma)$$

and π: $A \to x \mid \alpha\_\beta \in P$, then

$$\Gamma' = (N', B', L', <_c', l') \in \text{CSSG}(G),$$

where

(a) $N' = N \cup \{m_0, m_1, \ldots, m_k\}$ if $k = \mid x \mid$ and $N \cap \{m_0, \ldots, m_k\} = \varnothing$,

(b) $l'(m_0) = \pi$, $l'(m_i) = a_i$ $(1 \leqslant i \leqslant k)$ if $x = a_1 \cdots a_k$,

and $l'(n) = l(n)$ for all $n \in N$,

(c) if the B-most members of N are written in L order as

$$c_1, \ldots, c_p, d_1, \ldots, d_q, e, f_1, \ldots, f_r, g_1, \ldots, g_s,$$

where $p = \mid u \mid$, $q = \mid \alpha \mid$, $r = \mid \beta \mid$, $s = \mid v \mid$, then set

(i) $B' = [B \cup \{(m_i, m_0) \mid 1 \leqslant i \leqslant k\} \cup \{(m_0, e)\}]^+$

(ii) $L' = [L \cup \{(m_i, m_{i+1}) \mid 1 \leqslant i < k\} \cup \{(m_i, x) \mid 0 \leqslant i \leqslant k; (e, x) \in L\} \cup \{(x, m_i) \mid 0 \leqslant i \leqslant k; (x, e) \in L\}]^+$

(iii) $<_c' = <_c \cup \{(m_0, d_i) \mid 1 \leqslant i \leqslant q\} \cup \{(m_0, f_i) \mid 1 \leqslant i \leqslant r\}$.

Using either intuition or a straightforward but tedious inductive proof, it can be verified that for any CSSG $\Gamma = (N, B, L, <_c, l)$, (N, B, L, l) is a labelled DOG tree with domain S and codomain as in the implied derivation sequence. The construction of (N, B, L, l) is just that of a context-free derivation tree in the CFG that is the basis of the SCSG. For complete details, see Alter and Hart (1979), where the proof is given for type 0 grammars. $<_c$ then shows the contextual relationship.

Given a CSSG, it is now necessary to define the "free" relation, $<_f$, which shows that the application of a particular production is dependent upon all context uses of the symbol that it rewrites. First, let B^0 be the "immediate" below relation as defined previously. Then $<_r = B^0$, where $<_r$ is the rewrite dependency of Section 2. If $(a, b) \in <_r$ with $l(b) = A \in V - \Sigma$ and $l(a) = \pi\colon A \rightarrow x \mid \alpha\_\beta \in P$, we wish to have $(a, c) \in <_f$ whenever $(c, b) \in <_c$. The L relation is also to be adjusted to give $\vdash\!\!\!-$, the "reduced left-of" relation.

DEFINITION 3.5. Let $G = (V, \Sigma, P, S)$ be a SCSG with $\Gamma = (N, B, L, <_c, l) \in$ CSSG(G). The *free-relation* of Γ, denoted $<_f(\Gamma)$ or simply $<_f$, is:

$$<_f = \{(a, c) \mid \exists b \in N \quad \text{with} \quad (c, b) \in <_c \quad \text{and} \quad (a, b) \in <_r)\}.$$

The *total dependency relation*, $<$, is:

$$< = (<_r \cup <_c \cup <_f)^+.$$

The *reduced left-of relation*, $\vdash\!\!\!-$, is:

$$\vdash\!\!\!- = L - (< \cup <^r).$$

The *total context graph*, TCG, of Γ is

$$\text{TCG}(\Gamma) = (N, <, \vdash\!\!\!-, l).$$

THEOREM 3.1. *Let $G = (V, \Sigma, P, S)$ be a SCSG with $\Gamma = (N, B, L, <_c, l) \in$ CSSG(G). Then TCG(Γ) is a (labelled) DOG.*

Proof. Show in steps that all the conditions for a DOG are satisfied.

(a) $<$ is transitive by definition.

(b) $\vdash\!\!\!- \subseteq L$, so $\vdash\!\!\!- \cap \vdash\!\!\!-^r = \varnothing$ (because $L \cap L^r = \varnothing$).

(c) $\vdash\!\!\!- \cap < = \varnothing$ by definition ($\Rightarrow \vdash\!\!\!-^r \cap <^r = \varnothing$). Likewise, $<^r \cap \vdash\!\!\!- = < \cap \vdash\!\!\!-^r = \varnothing$.

(d) $= \cup < \cup <^r \cup \vdash \cup \vdash^r = N \times N$. This is shown by the fact that (N, B, L) is a DOG, if $(a, b) \in B \cup B^r$, then $(a, b) \in < \cup <^r$, and if $(a, b) \in L \cup L^r$, either $(a, b) \in \vdash \cup \vdash^r$ or $(a, b) \in < \cup <^r$.

(e) $< \cap <^r = \varnothing$. Assume this property (the TCG is acyclic with respect to $<$) holds for some CSSG, Γ, and note that the construction in Step 2 of Definition 3.4 together with the definition of $<'_f$ assures that $<' \cap <'^r = \varnothing$.

(f) $\vdash$ is transitive. Assume as an inductive hypothesis that this holds for the CSSG Γ in Step 2 of Definition 3.4 (the basis is easily established). Now, suppose $(a, b) \in \vdash'$ and $(b, c) \in \vdash'$. Certainly $(a, c) \in L'$ so $(a, c) \notin \vdash'^r$. Suppose $(a, c) \notin \vdash'$, so (by Part d above) the only possibility is that $(a, c) \in <' \cup <'^r$. Without loss of generality, assume that $(a, c) \in <'$. Since $<'$ adds nothing new to $N \times N$, we must have $a \in \{m_0, ..., m_k\} = M$. By the construction, nothing is lost by assuming that $a = m_0$. Since $(m_0, b) \in L'$, $(e, b) \in L$, and, of course, $(b, c) \in L$. For $(a, c) \in <'$, there must be some $x \in N$ with $(x, c) \in <$, $(m_0, x) \in <'$ and one of the following properties:

(i) $(x, e) \in <_c$ (hence (m_0, x) is a new element of $<'_f$).

Then $(x, b) \in <$ since $(e, b) \in \vdash$, $(b, c) \in \vdash$, $(x, e) \in <$, and $(x, c) \in <$ and any other possibility for (x, b) leads to a contradiction.

(ii) $(m_0, x) \in <'_c$. The situation is as above except that $(x, e) \in \vdash \cup \vdash^r$, again leading to contradictions unless $(x, b) \in <$.

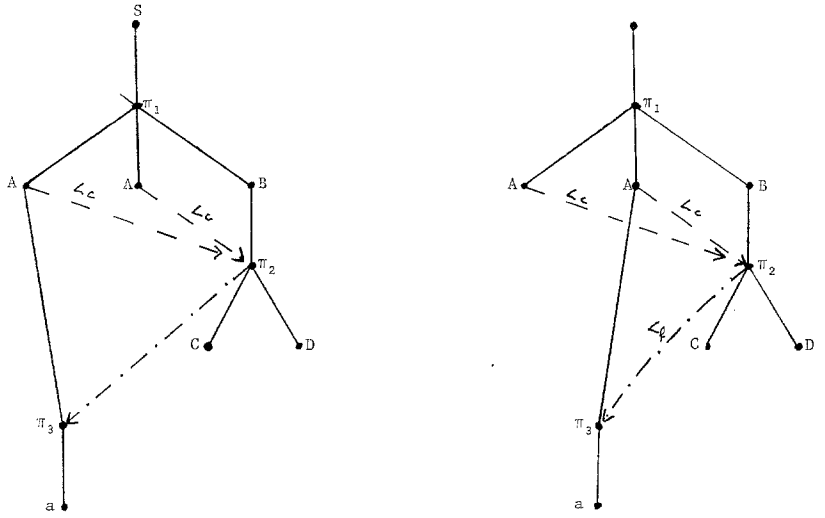
In either case $(m_0, b) \in <'$, a contradiction. All of the requirements for a DOG have been shown, completing the proof. Q.E.D.

4. Canonical Derivations in Strictly Context-Sensitive Grammars

In dealing with type 0 phrase structure grammars, the syntactical graph is a DOG, and the node labels read in the order of the POT give sufficient information to reconstruct the syntactical graph. More precisely, suppose $\Gamma = (N, B, L, l)$ is a labelled DOG with $N = \{n_1, ..., n_k\}$ such that $\rho(n_i) = i (1 \leqslant i \leqslant k$, ρ is the POT of $\Gamma)$. Then, the *trace of* Γ is $l(n_1 n_2 \cdots n_k)$. If Γ is a syntactical graph, the trace is the *derivation word* (Hart, 1976), and there is an effective one-to-one correspondence between derivation words and syntactical graphs of a given type 0 grammar. The subject of derivation words is explored in greater depth in (Hart, 1975 and Révész, 1977).

For SCSG's, however, the trace of the TCG is not unique. For instance, consider a grammar with productions:

$$\pi_1 : S \rightarrow AAB$$
$$\pi_2 : B \rightarrow CD \mid AA\_$$
$$\pi_3 : A \rightarrow a.$$



a) CSSG, α_1
cod(α_1)=aACD
(with respect to B and L)

trace(α_1)=trace(α_2)=Sπ_1AAB$\pi_2\pi_3$aCD
(with respect to < and $\vdash$)

FIG. 5. Two CSSG's with the same trace and the same TCG's.

Figure 5 shows two CSSG's of this grammar, both of which have the same trace.

As an additional example, the CSSG of Fig. 4 has trace

$$S\pi_1 AB\pi_3\pi_2 XYDEC\pi_4 AA\pi_5 FG.$$

As another contrast with derivation words, note that, in general, the codomain of the TCG is not the codomain of the DOG tree (N, B, L, l) since some B-most nodes may be used for context. This can be seen in Fig. 4 by deleting the application of the π_5 rule.

The value of the trace of the TCG lies in the fact that it allows us to define a meaningful "left-most" or "canonical" derivation for SCSG's. First, given a SCSG, $G = (V, \Sigma, P, S)$, a *derivation sequence* (in G) is a sequence

$$y = u_1\alpha_1 A_1\beta_1 v_1 \overset{\pi_1}{\Rightarrow} u_2\alpha_2 A_2\beta_2 v_2 \overset{\pi_2}{\Rightarrow} \cdots u_n\alpha_n A_n\beta_n v_n \overset{\pi_n}{\Rightarrow} u_n\alpha_n x_n\beta_n v_n = z,$$

where $n \geqslant 0$ and each $\pi_i : A_i \to x_i \mid \alpha_i\_\beta_i \in P$. The derivation is said to be from y to z.

Also, at each step,

$$u_{i+1}\alpha_{i+1}A_{i+1}\beta_{i+1}v_{i+1} = u_i\alpha_i x_i\beta_i v_i .$$

From Definitions 3.4 and 3.5 we see that each derivation sequence yields a TCG (or CSSG).

DEFINITION 4.1. Let $G = (V, \Sigma, P, S)$ be a SCSG. Two derivations are said to be *equivalent* if they yield the same CSSG (hence, the same TCG).

The derivation sequence at the beginning of Section 2 (see Fig. 4 as well) applies the productions in the order

$$\pi_1\pi_3\pi_4\pi_5\pi_2 .$$

There are two other derivation sequences equivalent to this one, namely those in which the order of rule application is

$$\pi_1\pi_3\pi_2\pi_4\pi_5 ,$$

and

$$\pi_1\pi_3\pi_4\pi_2\pi_5 .$$

This particular derivation is a member of an equivalence class of size three. The canonical (leftmost) member of a class of equivalent derivations can now be defined.

DEFINITION 4.2. *Let* $G = (V, \Sigma, P, S)$ be a SCSG with

$$u_1\alpha_1A_1\beta_1v_1 \overset{\pi_1}{\Rightarrow} u_2\alpha_2A_2\beta_2v_2 \overset{\pi_2}{\Rightarrow} \cdots \overset{\pi_n}{\Rightarrow} u_n\alpha_nx_n\beta_nv_n$$

a derivation sequence in $G(n \geqslant 0)$. This sequence is said to be *canonical* (leftmost) if and only if, for each i, $1 \leqslant i \leqslant n-1$, either

$$| u_{i+1}\alpha_{i+1}A_{i+1}\beta_{i+1} | > | u_i\alpha_i |,$$

or

$$| u_{i+1}\alpha_{i+1} | \geqslant | u_i |$$

This definition should be contrasted with the context-free leftmost derivations and the more general type 0 canonical derivation in which each derivation step is of the form

$$u_iv_ix_i \overset{\pi_i}{\Rightarrow} u_iw_ix_i = u_{i+1}v_{i+1}x_{i+1} ,$$

where $\pi_i : v_i \to x_i \in P$. The canonical requirement is then (Hotz, 1966; Griffiths, 1968; Hart, 1975, and elsewhere)

$$| u_i | < | u_{i+1}v_{i+1} |.$$

The reader can verify that of the three equivalent sequences (mentioned after Definition 4.1) the unique canonical derivation is $\pi_1\pi_3\pi_2\pi_4\pi_5$. Also, note that this is exactly the order in which the production names occur in the trace of the CSSG of this class of sequences. This observation is true in general, as is shown in the discussion after the following algorithm.

ALGORITHM 4.1. Construction of a canonical derivation sequence of an equivalence class of derivation sequences.

Input: A SCSG, $G = (V, \Sigma, P, S)$ and a CSSG,

$$\Gamma = (N, B, L, <_c, l).$$

Output: A canonical derivation sequence equivalent to all those with CSSG, Γ.

Algorithm. Construct $\mathrm{TCG}(\Gamma) = (N, <, \vdash, l)$ and ρ the POT of $\mathrm{TGC}(\Gamma)$. Let $K = \{k_1, \ldots, k_n\} = l^{-1}(P)$ be the set of nodes of Γ labelled by production names ordered so that $\rho(k_i) < \rho(k_{i+1})(1 \leqslant i \leqslant n)$. If $n = 0$, the length of the sequence is 0 and the algorithm halts. Otherwise, if $n > 0$, perform the following steps.

(1) Set $i = 1$, set $\Gamma_1 = (\{a_1, \ldots, a_r\}, \phi, L_1, \phi, l_1)$ as in Step 1 of Definition 3.4, where $\{a_1, \ldots, a_r\} = N_1$ are the B^r-most members of N listed in L order (that is, $a_i L a_{i+1}$ for $1 \leqslant i < r$). $B_1 = <_{c1} = \varnothing$.

(2) (a) Given $\Gamma_i = (N_i, B_i, L_i, <_{ci}, l_i)$ a CSSG in G, let e be the unique member of N_i such that $(k_i, e) \in B^0$.

(b) Write the B_i-most elements of N_i in L order as

$$c_1, \ldots, c_p, d_1, \ldots, d_q, e, f_1, \ldots, f_r, g_1, \ldots, g_s,$$

where $l(k_i) = \pi_i : A_i \to x_i \mid \alpha_i \beta_i$, $q = |\alpha_i|$, and $r = |\beta_i|$, and, in fact,

$$(k_i, d_j) \in <_c \qquad (1 \leqslant j \leqslant q),$$

and

$$(k_i, f_j) \in <_c \qquad (1 \leqslant j \leqslant r)$$

and there are no other elements $x \in N_i$ with $(k_i, x) \in <_c$.

(c) Set

$$u_i = l(c_1 \cdots c_p)$$

$$\alpha_i, A_i, \beta_i \text{ as required by } \pi_i = l(k_i),$$

and

$$v_i = l(g_1 \cdots g_s)$$

(3) Create $\Gamma_{i+1} = (N_{i+1}, B_{i+1}, L_{i+1}, <_{c,i+1}, l_{i+1})$ as in Step 2 of Definition 3.4 (where the notation of Step 2 above corresponds to that of Definition 3.4).

(4) $i = i + 1$; if $i < n$, go to Step 2.

(5) *Halt* with the canonical sequence indicated by the values of u_i, α_i, A_i, β_i, v_i, and π_i shown above.

Clearly, the constructed derivation sequence has Γ for a CSSG. We also need to show that the sequence is canonical. To see this, note that, for $1 \leqslant i < n$,

$$\rho(k_i) < \rho(k_{i+1})$$

so $(k_i, k_{i+1}) \in \vdash \cup <^r$. Consider the two cases

(a) $(k_i, k_{i+1}) \in \vdash$. By the construction, there is no other $k' \in K$ such that (k_i, k'), $(k', k_{i+1}) \in \vdash$. Also, since $(k_i, k_{i+1}) \notin <^r \cup <$, $l(k_{i+1})$, as a production, can only rewrite one of the nodes $f_1, \ldots, f_r, g_1, \ldots, g_s$ and must also take all of its context from these nodes. Hence

$$| u_{i+1} | > | u_i \alpha_i x_i |$$

implying the weaker condition,

$$| u_{i+1} \alpha_{i+1} A_{i+1} \beta_{i+1} | > | u_i \alpha_i |.$$

(b) $(k_i, k_{i+1}) \in <^r$. Thus, $l(k_{i+1})$ either rewrites one of the context symbols (at one of the nodes $d_1, \ldots, d_q, f_1, \ldots, f_r$) or else rewrites or uses for context one of the symbols of x_i (where $\pi_i : A_i \rightarrow x_i \mid \alpha_i \_ \beta_i$). Hence

$$| u_{i+1} \alpha_{i+1} | \geqslant | u_i |.$$

Therefore, considering the two cases shows that the sequence generated by the algorithm is canonical.

Finally, the canonical sequence is unique (i.e., only one canonical sequence can be constructed from a CSSG). At any point in the construction of a sequence from Γ, there is a set $K' = \{k'_1, \ldots, k'_t\} \subseteq K (t < n)$ of $<^r$-most members of K which have not been applied (i.e., $<^r$-most with respect to K). Suppose they are written with $k'_j \vdash k'_{j+1}$ so that the canonical sequence constructed in the algorithm applies the production $l(k'_1)$. If any other sequence is constructed from the graph, there must be some such point in its construction (say the creation of π_i, u_i, etc.), where some production $l(k'_j) (j \geqslant 2)$ is applied followed immediately by the application of $l(k'_1)$ as π_{i+1}. Now, $(k'_1, k'_j) \in \vdash$, so we must have the two inequalities:

$$| u_{i+1} \alpha_{i+1} A_{i+1} | \leqslant | u_i | \Rightarrow | u_{i+1} \alpha_{i+1} | < | u_i |,$$

and

$$| u_{i+1}\alpha_{i+1}A_{i+1}\beta_{i+1} | \leqslant | u_i\alpha_i |.$$

Therefore, the resulting sequence cannot be canonical.

Conclusion. There is exactly one canonical derivation sequence for every equivalence class of derivations in a SCSG. The order of rule application in this sequence is exactly the same as the order of appearance of production names in the TCG trace.

In order to construct the canonical sequence equivalent to some derivation sequence, first construct the CSSG and then perform Algorithm 4.1.

It is worth pointing out that the canonical sequence is constructed using all of the relations B, L, $<$, and $\vdash$. The pair of relations (B, L) is not sufficient by itself (they just show the underlying context-free derivation) nor is the pair $(<, \vdash)$ sufficient. To see this, consider Fig. 5. The two graphs represent distinct canonical derivations, but the TCG's are identical.

Finally, the concept of canonical and equivalent derivations presented here can be compared with similar concepts given by Buttelmann (1975). Buttelmann refers to the "strictly context-sensitive" grammars of this paper as "phrase structure" grammars, and he defines a concept of a righmost derivation in these grammars. This rightmost derivation is obtained by a complex sequence of interchange operations on a syntactical structure which is essentially the derivation sequence. The resulting derivation is clearly the same as the canonical derivation of this paper, as are the two concepts of equivalence. Buttelmann also has a notion of a type of graph (called "phrase structures") to represent these derivations. There are several important distinctions to be made between the two approaches however:

(1) Our CSSG's are much more compact than the "phrase structures" of Buttelmann, and they do not require the considerable amount of labelling used in the phrase structures.

(2) CSSG's show all of the dependencies involved in a derivation and they clearly show the underlying context-free derivation tree.

(3) CSSG's allow for direct computation of the canonical sequence from the TCG, which is a graph, and, in fact, the canonical sequence used here is not defined by Buttelmann. There is no need to work out the sequence of interchange operations to obtain the rightmost derivation.

5. Monotonic, Strict CS, and Mixed Grammars

The example in Section 1 showed the advantage of strict CS productions over their monotonic form in avoiding unnecessary ambiguity in the grammar. On

the other hand, a monotonic production $\pi : \alpha \to \beta$ (with $|\alpha| \leqslant |\beta|$) is far simpler than its SCS equivalent set of productions.

The weak equivalence of the monotonic and SCS forms is well known, and there are several techniques for converting a monotonic production to a set of SCS productions. For instance, consider the monotonic production

$$\pi\colon X_1X_2 \to Y_1Y_2Y_3\,,$$

(where the X's and Y's are individual symbols). Applying Salomaa's conversion (1973, p. 83) leads to the weakly equivalent set of SCS productions

$$\begin{aligned}(\pi, 1)&\colon X_1 \to Z_1 \,|\_X_2\\ (\pi, 2)&\colon X_2 \to Z_2Y_3 \mid Z_1\_\\ (\pi, -1)&\colon Z_1 \to Y_1 \,|\_Z_2Y_3\\ (\pi, -2)&\colon Z_2 \to Y_2 \mid Y_1\_Y_3\end{aligned}$$

(Z_1 and Z_2 are new symbols, unique to production π). Note that these four rules, if applied, will lead to a canonical derivation with the productions used in the order they are listed above.

Not only are these rules more complex (try drawing the TCG), but strong (structural) equivalence between derivations in the monotonic and SCS grammars is not preserved in any meaningful sense. To see this, suppose there is also a production (in the monotonic grammar)

$$\rho\colon X_0Y_1 \to U_1U_2U_3$$

and consider the derivation from $X_0X_1X_2$ to $U_1U_2U_3Y_2Y_3$ using π and ρ in sequence. In the corresponding (canonical) derivation sequence in the (weakly) equivalent SCSG, all of the four productions corresponding to ρ will be applied *after* $(\pi, -1)$ and *before* $(\pi, -2)$ in the sequence corresponding to π. Thus, the productions corresponding to π are not applied as a unit and structural equivalence is destroyed. The situation here is analogous to some of those analyzed more formally (in a categorical framework) by Benson (1977).

In conclusion, both SCS rules and monotonic rules have advantages over each other, depending upon the situation. In light of the fact that the SCS rules can prevent ambiguity, it would be sensible to allow more general rules of the form

$$\pi\colon u \to v \mid \alpha\_\beta, \qquad \text{where} \quad 1 \leqslant |u| \leqslant |v|$$

This leads to the definition

DEFINITION 5.1. A *mixed context-sensitive grammar* (MCSG) is a system

$$G = (V, \Sigma, P, S),$$

where V, Σ, and S have the usual meaning and each production in P is of the form

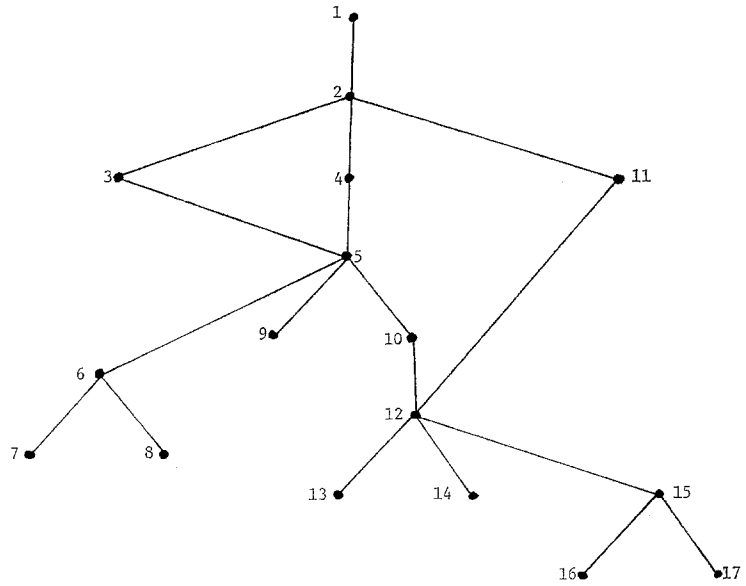
$$\pi\colon u \to v \mid \alpha\_\beta \qquad \text{where} \quad |u| \leqslant |v|, \ \alpha, \beta \in V^*, \qquad \text{and} \qquad u \in V^+ - \Sigma^+.$$

Derivations structures and canonical derivations in a MCSG can now be defined by a straightforward extension of the techniques in this paper. The use of MCSG production rules allows for the maximum efficiency in preventing ambiguity, preserving strong equivalence, and simplifying derivation structures.

6. Other Aspects of Context-Sensitive Derivations

In this chapter, we consider a number of other topics which arise in the study of CSSG's. First, it is shown that DOG's are actually permutations, thus allowing a description of CSSG's by means of permutations.

Let $K = \{1,\ldots, k\}$ with $P\colon K \to K$ a permutation of K (i.e., a one-to-one function). Then, we can create a DOG over K with relations B and L defined by:



$$P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 17 & 16 & 13 & 14 & 12 & 3 & 1 & 2 & 4 & 11 & 15 & 10 & 5 & 6 & 9 & 7 & 8 \end{pmatrix}$$

Fig. 6. The TCG of Fig. 4 represented as a permutation.

(1) $(i, j) \in L$ iff $i < j$ and $P(i) < P(j)$

(2) $(i, j) \in B$ iff $i > j$ and $P(i) < P(j)$

This DOG has nodes numbered according to its pre-order traversal and the value of the permutations is determined by the reverse of the post-order traversal. Figure 6 shows the TCG of Fig. 4 with the nodes numbered by the POT together with the corresponding permutation. (Only the transitive reduction is shown.) It should be clear from this that every DOG can be represented as a permutation (is isomorphic to a permutation).

Therefore, we could represent a CSSG compactly by a system (N, ρ_1, P_1, ρ_2, P_2, l), where N is the set of nodes, ρ_1 is the POT of (N, B, L), P_1 is the permutation of (N, B, L) (using the node numbering defined by ρ_1), and ρ_2, P_2 play the same role for (N, $<$, $\vdash$). l is the node labelling function.

Next, we could develop by the obvious but tedious methods definitions for the juxtaposition (side by side application) and composition (successive application) of two derivations (actually, classes of derivations as represented by CSSG's). Although such definitions would be dull and unrevealing for the graphs, they could be interesting if derivations were represented by permutations as suggested above.

Finally, a parsing algorithm for SCSG's could be developed by a two stage process.

(1) Parse the input according to the underlying CFG. If this grammar is, say, $LR(k)$, then at most one derivation will exist, giving the B and L relations of the CSSG. In general, however, a more powerful technique would be required to create a set of context-free parses. The Cooke–Younger–Kasami or Early algorithms (Aho and Ullman, 1972, pp. 314ff) are examples of such techniques.

(2) An algorithm is then needed which, for each potential context-free parse (B and L relations of a derivation tree), would create an admissable $<_c$ relation. Unfortunately, the problem of determining such $a <_c$ relation is known to be *NP*-complete (Stradel, 1979).

Stradel's result really is not surprising, for it is well known that the problem of membership in an arbitrary context-sensitive language (represented by a linearly bounded automaton or a monotonic grammar) is complete for P-space (Karp, 1972).

RECEIVED: April 13, 1979; REVISED: January 29, 1980

REFERENCES

AHO, A. V. AND ULLMAN, J. D. (1972), "The Theory of Parsing, Translation, and Compiling," Vol. I: "Parsing," Prentice–Hall, Englewood Cliffs, New Jersey.

ALTER, R. AND HART, J. M. (1979), Enumerating syntactical graphs and lattices of derivations, *Internat. J. Comput. System Sci.* **8**, 261–277.

BENSON, D. B. (1977), Some preservation properties of normal form grammars, *SIAM J. Comput.* **6**, 381–402.

BUTTELMANN, H. W. (1975), On the syntactic structures of unrestricted grammars, I. Generative grammars and phrase structure grammars, *Inform. Contr.* **29**, 29–80.

EICKEL, J. AND LOECKX, J. (1972), The relation between derivations and syntactical structures in phrase structure grammars, *J. Comput. System Sci.* **6**, 267–282.

GRIFFITHS, T. V. (1968), Some remarks on derivations in general rewriting systems, *Inform. Contr.* **12**, 27–54.

HARRISON, M. A. (1978), "Introduction to Formal Language Theory," Addison–Wesley, Reading, Mass.

HART, J. M. (1975), Derivation languages and syntactical categories, *Inform. Contr.* **28**, 204–220.

HART, J. M. (1976), The derivation language of a phrase structure grammar, *J. Comput. System Sci.* **12**, 64–79.

HART, J. M. (1979), Doubly Keyed Lists for Binary Relations, *in* "Proceedings, 1979 Conference on Information Sciences and Systems, March 28–30, The Johns Hopkins University, Baltimore, Maryland."

HOTZ, G. (1966), Eindeutigkeit und Mehrdeutigkeit formaler Sprachen, *Elektron. Informationsverarbeit. Kybernetik.* **2**, 235–246.

JOSHI, A. K. AND LEVY, L. S. (1977), Constraints on structural descriptions: local transformations, *SIAM J. Comput.* **6**, 272–284.

KARP, R. M. (1972), Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–104, Plenum, New York.

PETERS, P. S. AND RITCHIE, R. W. (1969), Context sensitive immediate constituent analysis—context free languages revisited, *in* "Proceedings, ACM Symp. Theory of Computing, Association for Computing Machinery, New York, 1969," pp. 1–10.

RÉVÉSZ, G. (1977), Algebraic properties of derivation words, *J. Comput. System Sci.* **15**, 232–240.

SALOMAA, A. (1973), "Formal Languages," Academic Press, New York/London.

STRADEL, M. (1978), Die Zeitkomplexität des Normalisierungsproblems bei kontextsensitiven Grammatiken, *Acta Inform.* **9**, 309–329.